⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平1－128136

⑤1Int.Cl.⁴	識別記号	庁内整理番号	⑬公開　平成1年(1989)5月19日
G 06 F 9/44	322	F－8724－5B	
7/544		7056－5B	

審査請求　未請求　発明の数 1　（全7頁）

⑤4発明の名称　コンパイル処理方式

⑳特　願　昭62－286604
㉒出　願　昭62(1987)11月13日

⑦2発明者　坂本　真理子　神奈川県川崎市中原区上小田中1015番地　富士通株式会社内
⑦2発明者　長倉　浩士　神奈川県川崎市中原区上小田中1015番地　富士通株式会社内
⑦2発明者　高嶋　秀夫　神奈川県川崎市中原区上小田中1015番地　富士通株式会社内
⑦1出願人　富士通株式会社　神奈川県川崎市中原区上小田中1015番地
⑦4代理人　弁理士　森田　寛　外2名

明　細　書

1．発明の名称

コンパイル処理方式

2．特許請求の範囲

数学関数を含むソースプログラム（9）をコンパイルするコンパイル処理方式において，

前記ソースプログラム（9）の解析を行う構文解析部（3）と，

前記数学関数の使用形態を認識して別の数学関数に変換可能か否かを判断する認識部（5）と，

前記数学関数を前記別の数学関数に変換する変換部（6）と，

前記ソースプログラム（9）に対応する目的コードを生成する目的コード生成部（7）とを少なくとも備え，

前記変換が可能である場合，前記構文解析部（3）における前記ソースプログラム（9）の解析結果について，少なくとも前記数学関数を前記別の数学関数に変換し，この変換結果について目的コードを生成することによって，前記数学関数の処理を最適化する

ことを特徴とするコンパイル処理方式。

3．発明の詳細な説明

〔概要〕

例えば平方根関数のような数学関数を含むソースプログラムのコンパイルを行うコンパイル処理方式に関し，

数学関数がより高速処理ができる別の数学関数に変換可能か否かを認識し，その変換を行うことによって，数学関数の処理を最適化することを目的とし，

数学関数を含むソースプログラムをコンパイルするコンパイル処理方式において，前記ソースプログラムの解析を行う構文解析部と，前記数学関数の使用形態を認識して別の数学関数に変換可能か否かを判断する認識部と，前記数学関数を前記別の数学関数に変換する変換部と，前記ソースプ

ログラムに対応する目的コードを生成する目的コード生成部とを少なくとも備え，前記変換が可能である場合，少なくとも前記数学関数を前記別の数学関数に変換し，前記数学関数の処理を最適化するように構成する。

〔産業上の利用分野〕

本発明はコンパイル処理方式に関し，更に詳しくは，例えば平方根関数のような数学関数を含むソースプログラムのコンパイルを行うコンパイル処理方式に関する。

科学技術計算の分野で用いられるプログラムにおいては，基本的な数学関数（平方根関数，指数関数，対数関数，三角関数等）が大きなコストを占める（実行時間の大きな割合を占める）ものが少なくない。

このようなプログラムでは，数学関数の処理の最適化による処理速度の高速化が，実行性能の向上つまりプログラム全体の実行時間の大幅な短縮に大きな効果をもたらす。

〔従来の技術〕

第4図は従来の数学関数のコンパイル処理の例を示す図である。

ソースプログラムが入力されると（①），コンパイラはその構文解析を行い（②），その解析結果に従ってソースプログラムと等価の中間言語（コード）を生成する（③）。

ソースプログラムがＦＯＲＴＲＡＮにより図示の如く，

Ｆ＝Ｑ／ＳＱＲＴ（Ｘ＋Ｙ） ――― （１）

と記述されていた場合，中間言語は例えば，

$$\left.\begin{array}{l} t1 = X + Y \\ t2 = SQRT(t1) \\ F = Q / t2 \end{array}\right\} \quad (2)$$

とされる。ここで，ＳＱＲＴは平方根関数，ｔ１とｔ２はコンパイラが生成した作業変数である。

次に，コンパイラは，生成された中間言語に従って，目的（オブジェクト）コードを生成する（④）。従って，数学関数を処理するためのプロ

グラムが格納された関数ライブラリからは，平方根関数ＳＱＲＴのプログラムが呼出される（⑤）。

平方根関数ＳＱＲＴの処理においては，（２）式のｔ２＝ＳＱＲＴ（ｔ１）を処理する場合，実際は，計算機内ではその処理の都合上，

$$\left.\begin{array}{l} t2' = 1 / \sqrt{t1} \\ t2 = 1 / t2' \end{array}\right\} \quad (3)$$

の如き処理が行われるようにされている。

従って，（１）式は，実際は，

Ｆ＝Ｑ／（１／（１／$\sqrt{ｔ１}$））――（４）

の如く処理される。

〔発明が解決しようとする問題点〕

前述の従来技術においては，数学関数の処理の最適化の手段として，主に関数自体（関数ライブラリに格納された関数プログラム）の性能向上のみが考慮され，数学関数がソースプログラムにおいてどのように使用されているかという使用形態が考慮されることはなかった。

即ち，（３）式の如き処理のなされる数学関数ＳＱＲＴが（１）式の如くソースプログラムで使用される場合，

Ｆ＝Ｑ＊ｔ２′＝Ｑ＊１／$\sqrt{ｔ１}$ ――（５）

の如くに変換することにより，（４）式に比べ２回の割算を省略して高速に処理することが可能であるにも拘らず，（５）式による処理が行われることはなかった。

換言すれば，数学関数の使用形態を認識して，数学関数を別の数学関数に変換する，即ち，関数ライブラリからの呼出し形式を変換する（（３）式のｔ２′＝１／$\sqrt{ｔ１}$ のみを呼出すか，またはこれに相当する関数プログラムを呼出す）ことは行われない。即ち，従来のコンパイラは，数学関数の使用形態の認識による最適化という手段を持たないため，ソースプログラムの記述そのままの数学関数に対応するような目的コードを生成するのみであった。

本発明は，数学関数がより高速処理ができる別の数学関数に変換可能か否かを認識し，その変換

を行うことによって，数学関数の処理を最適化することが可能なコンパイル処理方式を提供することを目的とする。

〔問題点を解決するための手段〕

第１図は本発明の原理構成図であり，本発明によるコンパイラを備えた処理装置を示している。

第１図において，１は中央処理装置（ＣＰＵ）とメモリとを含む処理装置，２はコンパイラ，３は構文解析部，４は最適化処理部，５は認識部，６は変換部，７は目的コード生成部，８は結合編集処理部，９はソースプログラム，１０はオブジェクトモジュール，１１は関数ライブラリ，１２は変換テーブル，１３はロードモジュールである。

ＦＯＲＴＲＡＮの如き高級言語で記述されたソースプログラム９は，コンパイラ２によって機械語コードに翻訳され，その翻訳結果である目的コードの集合はオブジェクトモジュール１０として格納される。リンケージエディタの如き結合編集処理部８は，目的コードに従って関数ライブラリ

このために，例えば，変換テーブル１２が用意される。変換テーブル１２には，変換のために満足すべき所定の条件と，変換後の別の数学関数およびその使用条件とが登録される。即ち，認識部５は，まず数学関数の使用形態を認識し，次にこれが変換テーブル１２に登録された所定の条件と一致するかを変換テーブル１２を参照して判断する（一致する時に変換可能とされる）。なお，変換テーブル１２は，関数ライブラリ１１内または最適化処理部４内に設けられてもよい。

変換部６は，認識部５が変換可能とした数学関数について，変換テーブル１２を参照することによって，他の数学関数への変換を行い，これに伴う必要な処理を行う。

関数ライブラリ１１は，変換前の数学関数についての関数プログラムの他，変換後の数学関数についての関数プログラムも含むようにされる。

〔作用〕

コンパイラ２の最適化処理部４は，数学関数の

１１から呼出した関数プログラムと目的コード（オブジェクトモジュール１０）とを結合させて，関数プログラムを直ちに参照できるようにした（実行可能な形式にした）ロードモジュール１３を編集し，格納する。

コンパイラ２において，構文解析部３は，ソースプログラム９を解析して，中間言語（最適化前）に展開する。最適化処理部４は，数学関数の使用形態を考慮することによって，中間言語の最適化を行う。この最適化された中間言語について，目的コード生成部７が目的コードを生成し，オブジェクトモジュール１０に展開する。

最適化処理部４においては，次の如き処理が行われる。

構文解析部３からの中間言語を入力とする認識部５は，その中間言語に含まれる数学関数がより高速処理できる別の数学関数に変換可能か否かを認識する。この認識は，数学関数が所定の条件を満足する形で使用されているかを調べることによって行われる。

使用形態が所定の条件を満たす場合に，自動的に，別の数学関数に変換するというコンパイル処理を行う。また，この変換に伴う処理として，最適化処理部４は，別の数学関数をその使用条件に合わせて使用するために，中間言語の一部を変換するコンパイル処理を行う。

これにより，ソースプログラム９との等価性を維持しつつも，ソースプログラム９における記述に拘らず，より高速処理が可能な別の数学関数を用いて記述されたロードモジュール１３が得られる。

このように，高速処理のための数学関数の最適化が成された結果，処理装置１においてロードモジュール１３（ソースプログラム９）を実行する場合，その実行時間を短縮することができる。

〔実施例〕

第２図は本発明の一実施例説明図であり，簡単な具体例を示している。

ソースプログラム９がＦＯＲＴＲＡＮで記述さ

れており，その中に図示の如く前述の（1）式と同様の文があったとする。

このソースプログラム9がコンパイラ2に入力されると，構文解析部3がこれを解析し，最適化前の中間言語からなる第1中間テキスト14を生成する。この時，（1）式は図示の如く前述の（2）式と同様に展開される。

第1中間テキストは最適化処理部4に入力され，最適化された中間言語からなる第2中間テキスト15が生成される。この時，（2）式は，

$$\left.\begin{array}{l} t1 = X + Y \\ t3 = XSQRT(t1) \\ F = Q * t3 \end{array}\right\} \quad (6)$$

の如く展開されている。ここでXSQRTは平方根の逆数を求める関数，t3はコンパイラ2が生成した作業変数である。また，関数SQRTとXSQRTとの関係は，変数をt4とすると，

$$\left.\begin{array}{l} XSQRT(t4) = 1/\sqrt{t4} \\ SQRT(t4) = 1/XSQRT(t4) \end{array}\right\} \quad (7)$$

の如く表される。通常，$\sqrt{x}$を求めるためには，

その変換部6において当該変換を第1中間テキスト14（式（2））に施して第2中間テキスト15（式（6））を得る。これにより，数学関数SQRTは，ソースプログラム9におけるその使用形態（割算の除数として用いられる）までを考慮して，最適化処理されたこととなる。

目的コード生成部7は，第2中間テキスト15が入力されると，これに基づいてオブジェクトモジュール10を展開する。

結合編集処理部8は，入力されたオブジェクトモジュール10に従って，数学関数XSQRTを関数ライブラリ11から呼出す。そして，オブジェクトモジュール10と数学関数とを結合編集し，ロードモジュール13を生成する。従って，ロードモジュール13は，ソースプログラム9の記述が数学関数SQRTであるにも拘わらず，より高速処理が可能な数学関数XSQRTを用いて記述された機械語コードからなるものとされる。

処理装置1がロードモジュール13を実行する場合，ソースプログラム9の記述にそのまま対応

ニュートン＝ラプソン法を用い，まず$1/\sqrt{x}$を求めてからその逆数をとる処理を行う。つまり，$1/\sqrt{x}$を求めるのであれば，本来，最後の逆数をとる処理は不要である。逆数をとる処理は1回の割り算で実現されるため，$\sqrt{x}$を求める関数SQRT(x)よりも$1/\sqrt{x}$を求める関数XSQRT(x)の方が割り算の処理が1回だけ少なくてすむ。

（2）式と（6）式とを比較すると，次のようである。変数t3を求める処理は，（7）式の関係を考慮すると，変数t2を求める処理よりも割算が1回少ないので，この分高速に処理される。また，Q＊t3の処理は，割算よりも掛算が高速処理できるので，Q／t2の処理よりもこの分速く処理できる。従って，（6）式は（2）式をより高速に処理することに適した形式に変換したものといえる。

このように，最適化処理部4は，その認識部5において数学関数SQRTがより高速処理できる数学関数XSQRTに変換できることを認識し，

したコンパイル処理（第1中間テキスト14を得る処理）に従ったロードモジュールを実行する場合に比べ，数学関数の処理の最適化により，その実行時間を短くすることができる。なお，このための処理は最適化処理部4において自動的に行われるので，プログラマの負担はなく，また，特殊な関数をソースプログラム中に導入してその可搬性を損なうこともない。

第3図は最適化処理部4における最適化処理フロー図である。

① 最適化処理部4に第1中間テキスト14が入力されると，認識部5は，使用する数学関数がSQRTであることを認識する。換言すれば，関数ライブラリ11から呼出す数学関数はSQRTであることを認識する。

② さらに，認識部5は，数学関数SQRTを用いた処理の結果の値（t2）が割算の除数として参照されていることを認識する。

③ 認識部5は，処理②で得た数学関数SQRTの使用形態について，これが数学関数SQRT

をより高速処理可能な別の数学関数に変換できる条件に適合しているか否かを判断する。このために，認識部５は変換テーブル１２を参照する。

変換テーブル１２には，数学関数ＳＱＲＴを用いた処理の結果の値（ｔ２）が割算の除数として参照されている時は，別の数学関数ＸＳＱＲＴに変換可能なことが登録されている。

④ 認識部５は，数学関数ＳＱＲＴを用いた処理の結果の値（ｔ２）がソースプログラム９（第１中間テキスト１４）において他に参照されているか否かを判断する。

認識部５において，次の

（ア）数学関数ＳＱＲＴが使用されている。

（イ）その処理結果の値ｔ２が割算の除数として用いられている。即ち，この使用形態が変換テーブル１２に登録されている。

（ウ）その処理結果の値ｔ２が他で参照されていない。

という命令列が認識された時，変換部６は次

数を含むソースプログラムのコンパイル処理において，数学関数の使用形態までを考慮することにより，当該数学関数をより高速処理可能な別の数学関数に変換することができ，プログラム全体の実行時間を短縮することができる。

４．図面の簡単な説明

第１図は本発明の原理構成図，

第２図は一実施例説明図，

第３図は最適化処理フロー図，

第４図は従来のコンパイル処理の例を示す図。

１は処理装置，２はコンパイラ，３は構文解析部，４は最適化処理部，５は認識部，６は変換部，７は目的コード生成部，８は結合編集処理部，９はソースプログラム，１０はオブジェクトモジュール，１１は関数ライブラリ，１２は変換テーブル，１３はロードモジュール，１４および１５は中間テキストである。

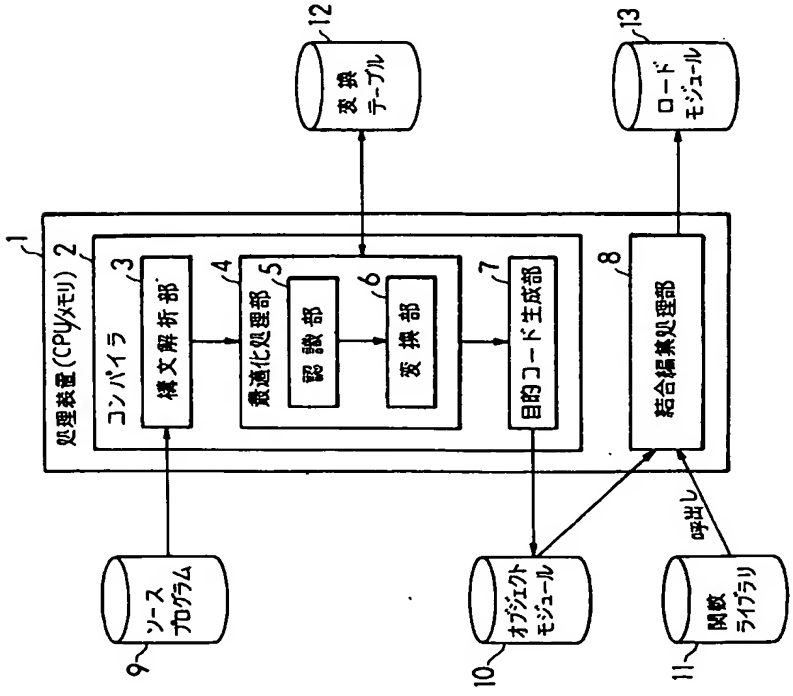
の処理を行う。

⑤ 変換部６は，変換テーブル１２を参照して数学関数ＳＱＲＴを別の数学関数ＸＳＱＲＴに変換する。即ち，（２）式中のｔ２＝ＳＱＲＴ（ｔ１）を，（６）式中のｔ３＝ＸＳＱＲＴ（ｔ１）に変換する。

⑥ さらに，変換部６は，変換テーブル１２を参照して，処理⑤のような変換を行った時の別の数学関数ＸＳＱＲＴの使用形態を知り，必要な処理を行う。即ち，数学関数ＳＱＲＴの処理結果の値ｔ２を除数として参照している割算（（２）式のＦ＝Ｑ／ｔ２）を，別の数学関数ＸＳＱＲＴの処理結果の値ｔ３を乗数として参照する掛算（（６）式のＦ＝Ｑ＊ｔ３）に変換する。これにより，（２）式と同じくソースプログラム９の（１）式に等しい（６）式が得られる。

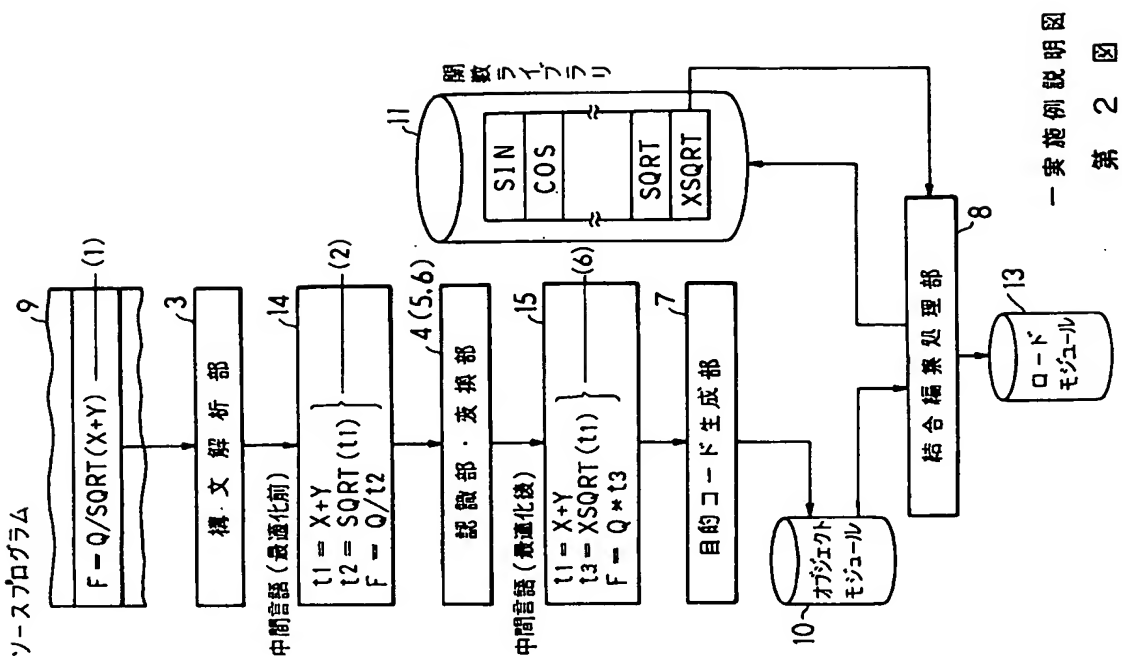
〔発明の効果〕

以上説明したように，本発明によれば，数学関



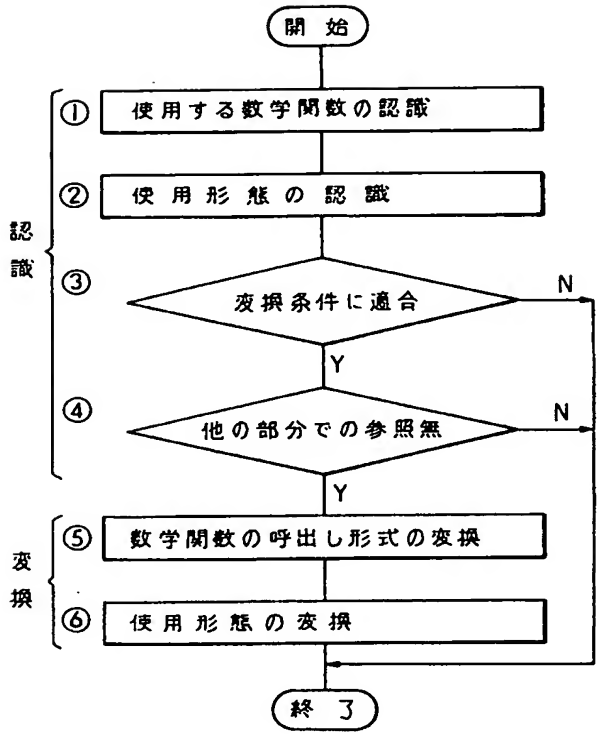
本発明の原理構成図

第 1 図



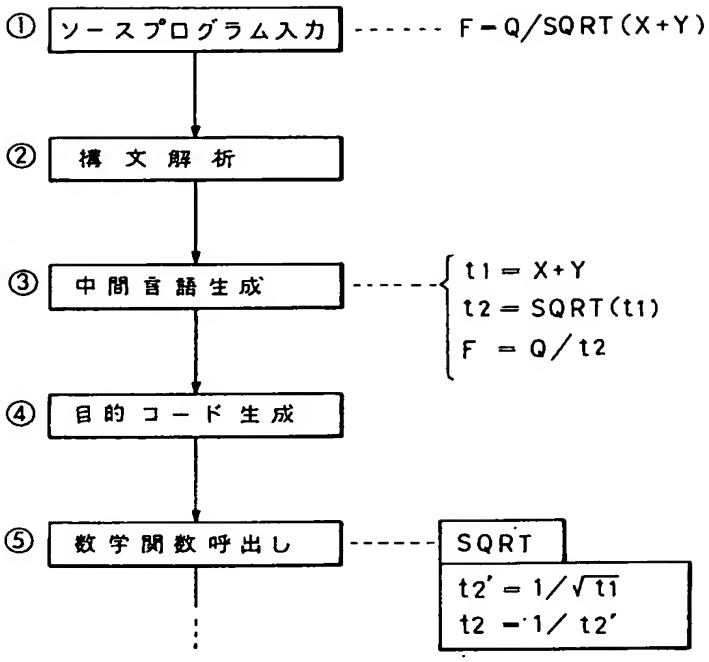
一実施例説明図

第 2 図



最適化処理フロー図

第 3 図



従来のコンパイル処理の例を示す図

第 4 図